

Performance Metrics for High End Computing

Jeffrey S. Vetter

Center for Applied Scientific Computing
LLNL
7000 East Ave
MS: L-560
Livermore, CA 94551
+1-925-424-6284
vetter3@llnl.gov

Theresa L. Windus

Molecular Science Software Group
PNNL
P.O. Box 999
MSIN: K8-91
Richland, WA 99352
509-376-4529
theresa.windus@pnl.gov

Brent Gorda

Future Technologies
LBNL
One Cyclotron Road
MS: 50A 1148
Berkeley, CA 94720
510-495-2493
bgorda@lbl.gov

Overview

This white paper addresses three separate questions in the HECRTF call for white papers: (2d) performance metrics that quantify benefits; (5) practical performance measures for system procurement that correlate well with realized performance of actual applications; and, (6) methods for deriving system performance targets from actual or projected application requirements or other user needs.

Introduction

Performance of HEC systems is a well-studied topic [1-3]; this paper does not attempt to rework this vast subject, but rather to draw attention to the important challenges for organizations that procure HEC systems. As Figure 1 illustrates, many factors influence the value of a high-end computer of which system performance is one component. For example, a major factor in most procurement activities is the total cost of ownership and the price/performance ratio. Nevertheless, this white paper focuses on only those factors that influence performance. Other HECRTF questions address additional factors, such as total cost of ownership.

Most metrics can be separated into three separate classes: application metrics, machine metrics, and performance metrics. Application metrics define the resource requirements of the application, independent of any architecture. Example application metrics include the number of floating-point operations, load instructions, amount of memory or disk required for an algorithm solution. Machine metrics are the specific capabilities and resources offered by a candidate computer system. Example metrics include peak floating-point operation and peak memory bandwidth rates. Other, perhaps more useful, metrics include sustained floating-point operations and sustained memory bandwidth, usually measured with microbenchmarks. Finally, performance metrics integrate application metrics with machine metrics to give an overall view of the performance of a specific application on a particular computer system.

In addition, metrics are valuable during all phases of the HEC system lifecycle: design, procurement, installation, and production. However, the selection and use of these metrics must balance two competing criteria: they must be practical to calculate and they must be flexible so that they can be used in a wide range of scenarios. Application metrics, for instance, are especially valuable in the design and

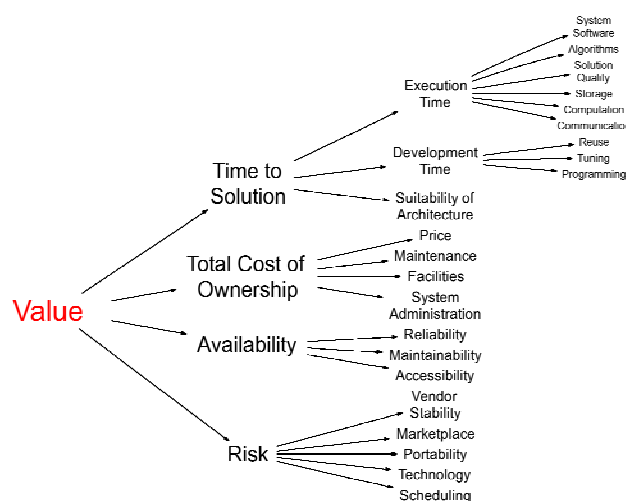


Figure 1: Factors that influence the value of a high end computing system.

procurement stages of the lifecycle because they can drive the selection of an appropriate platform. During installation and production use, performance metrics supply more detailed, diagnostic information that affords users the opportunity to optimize their applications for specific platforms.

Performance metrics that quantify benefits [2(d)]

The single unifying performance metric of high end computing is Time-To-Solution (TTS). TTS includes program development, setup, and execution time as necessary for the individual application to produce a solution for the customer. Some of the factors that influence TTS are system component characteristics, system architecture, programming environment, algorithm choice, and the suitability of the system to the application. In this framework and in addition to the traditional metric of execution time, TTS is inextricably tied to the development, and pre- and post-processing times for an application.

TTS is the single unifying metric of HEC; however, measuring the execution time and development time on future computer systems is currently not practical, and this degrades the usefulness of TTS as a metric. In many cases, performance expectations for a new procurement cannot be derived empirically, either because the basic components of the system do not exist, or they do not exist at the scale required for the final system. Few efficient and accurate techniques and tools exist for estimating the performance of a system prior to its existence. Once the target computer system is built at the scale required, users can empirically measure the execution time of their application. This is especially true for many government systems that represent the very first system of that type (serial number '1').

Development, Pre-, Post-processing time

Development time depends directly on the programmability and performance stability of the target system. Language, compilers, libraries, debuggers, performance tuning tools, and human programming skills directly impact TTS. Any time lost in development delays the ability of the application to provide a solution to the user. In addition, any pre- and post-processing activities, such as mesh generation, CAD design, and visualization also unequivocally impact TTS. The HEC community needs to better understand how development time influences the overall TTS. These activities impact TTS; however, they are outside the scope of this document so we will not address them further.

Execution time

There are several methods for measuring and reporting performance of a system [1-3]. Generally, users are interested either in reducing execution time of a specific task or in increasing throughput of a set of tasks. Execution time can be defined in several ways; we use the wall-clock or elapsed time, which is the time to complete a specific task including all sources of latency (e.g., I/O activities, OS overheads, memory accesses). The set of tasks used is called the system workload. For high-end applications, existing metrics such as relative speedup, parallel speedup, and parallel efficiency provide a basis for performance measurement, comparison, and evaluation [1].

There are generally five levels of workloads for evaluating HEC systems: real applications, scripted applications, kernels, toy benchmarks, and synthetic benchmarks. In selecting a workload, it is imperative that the selected workloads are representative, well defined, and scale to the size of computer system and input problem under consideration. For example, the workload should define configuration parameters such as input problem size, numerical tolerances, definition of the algorithm, and initial conditions; they must be clearly specified as part of the requirements for the benchmark. This restriction allows for a comparison of results on a particular machine and among different machines. It also allows for comparisons to be made between different applications that claim to use similar algorithms to solve a particular problem.


Each computational science area has its own set of metrics that are useful to that community and are generally meaningful only to those in the field. However, to the scientist, these metrics have the most meaning since they are directly related to the science that can be accomplished. For example, climate-modeling applications often use the number of years that can be simulated in an hour (throughput).

Whereas, in the field of chemistry, a molecular dynamics simulation of a biological system might use the wall clock time to simulate a nanosecond simulation with femtosecond time steps. Again, in each of these examples, the complete specification for the benchmark needs to be described and special conditions need to be noted.

In addition to TTS, users must have reasonable response time to their job submissions for their production jobs as well as their software development jobs. If a user's job stalls in a job queue for days in order to execute the application, then the effectiveness of that platform in the user's perspective is degraded. ESP, described later, is one such benchmark that can help evaluate the effectiveness of job response times.

Practical performance measures for system procurement that correlate well with realized performance of actual applications [5]

Numerous benchmarks correlate reasonably well with mission applications, however, this statement is entirely dependent on each code's respective computation and communication characteristics. Example benchmarks include LINPACK (used for the Top500), the SPEC benchmark suite, the NAS Parallel Benchmarks, and the Streams benchmark. It is well known that the LINPACK benchmark correlates positively with the theoretical peak of a platform and with applications that rely on dense-matrix calculations for the majority of their execution. However, LINPACK is not a reliable predictor of performance for applications that utilize other popular and necessary computational science tools, such as unstructured meshes, adaptively refined meshes, or sparse matrix solutions. Further, dense-matrix computations do not have a good correlation to applications utilizing contemporary software design such as modular design, object-oriented design, component technology, or other information hiding techniques. The peak performance metric is often not directly correlated to sustained performance for important applications; however, it is a practical and low risk specification that is measurable for most any computer system - past, present, future.

Several computer centers, such as the Molecular Sciences Computing Facility at PNNL  concentrate on a few scientific disciplines, and have a very detailed picture of many of the applications that are executed on their systems (i.e. what percentage of time is spent on each application and the individual models used in these applications). In these cases, it is possible to develop benchmarks that are representative of the workload and scalability using the types of TTS execution metrics discussed earlier. Generally, a throughput model is also developed that simulates a particular mix of applications and specifies exactly how many jobs must be in the batch queue at the same time, the percent usage of the machine that must be met, and the total time for these jobs to complete. This is, of course, a daunting task and requires extensive knowledge of the current job mix and a projection of the future job mix.

Making predictions of realized performance on yet-to-be-built computer systems is an inexact science at best. Few, if any, simulation and modeling tools provide both the requisite accuracy in performance prediction and the ability to efficiently analyze more than toy codes. Because of this, most procurements include requirements for other metrics to insure that the entire system provides a useable platform. For example, most DOE procurements include requirements for memory capacity, reliability, serviceability, and byte/flop ratios for memory, interconnect, and IO. Finally, it is often impractical to expect vendors to assemble a platform at scale for the sole purpose of running benchmarks for a procurement activity. In many cases the processors, compute nodes, or interconnect technology may not even be available for the performance testing. Thus, performance measures from a slightly different system must be suspect, and different from the actual performance measured on the final delivery system. To this end, the HEC community needs new, systematic performance measurement, modeling, and simulation tools to allow the analysis and evaluation of these metrics for both future and existing systems. Existing tools, such as cycle accurate simulators, can be inefficient, while high-level analytical modeling techniques can be difficult to create and validate for a wide range of applications and architectures.

Effective System Performance (ESP) Rating

A recent addition to procurement practices in the field is the Effective Systems Performance Rating, or ESP. ESP is designed to evaluate systems for overall effectiveness, independent of processor performance. The ESP test suite simulates “a day in the life of a production platform” by measuring system utilization under simulated real world conditions taking into account both hardware and system software performance. Developed by NERSC as part of a major system procurement process, ESP is designed to predict the effectiveness of a system before purchase, as well as to evaluate system changes before they are put into production use.

The goals of the ESP rating include: determining how well an existing system supports a particular scientific workload, assess systems for that workload before purchase, provide quantitative effectiveness information regarding system enhancements, compare different systems on a single workload or discipline, compare system-level performance on workloads derived from different disciplines, focus on real world system attributes like job launch and termination, and job scheduling for a workload with wildly varying CPU count and runtime requirements and system resiliency under reboot.

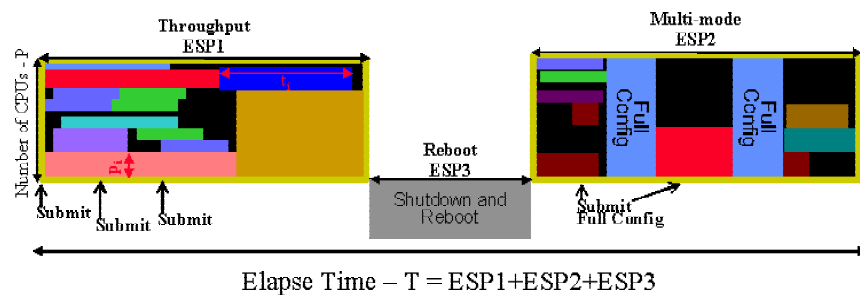


Figure 2: ESP Timeline

The Effective System Performance (ESP) is defined there as:

$$ESP = \frac{(AMT1 + AMT2)}{(ESP1 + ESP2 + ESP3)}$$

The three ESP components (throughput, multi-mode and reboot) absolute minimum times (AMT1 and AMT2) and observed ESP components (ESP1, ESP2 and ESP3) are defined in Figure 2. Given a total amount of work for each application, a hypothetical absolute minimum time, (AMT), can be computed by dividing the work by the system size. Since all components of ESP are positive and the numerator is always less than the denominator, $0 \leq ESP \leq 1$ (ESP values closer to unity are better).

Example: Suppose the proposed system has 12,288 CPUs and the scheduler is able to schedule the throughput workload at 95% utilization and the multi-mode workload at 90% utilization (quite possible with a fully integrated gang-scheduler utilizing parallel system initiated checkpoint/restart) and the measured ESP reboot time is 2 hours. Then $ESP = (3.00 + 3.06) / (3.00 / 0.95 + 3.06 / 0.90 + 2.00) = .71$. At a lower level than the ESP are design metrics and associated requirements that most procurement teams specify. Recent procurements, such as ASCI Purple, have included detailed metric requirements on byte/flop ratios for memory, interconnect, and IO. Additionally, with the scale of these new systems, contracts are growing to include requirements for facilities: power, cooling, floor space, etc.

Procurement/Design Metrics

Many procurement activities include a peak and sustained floating-point rate. Because of the risk involved in estimating TTS on future platforms, procurers and vendors alike gravitate toward peak floating point operations per second as a metric. Vendors can simply multiply the number of processors by the peak floating point rate of target processor to generate an exact measure for aggregate peak floating point rate. As mentioned earlier, virtually all of the other metrics here, including TTS, require that vendors project the running time of an application on a target architecture, which at the time of

award, may not be available because it has not been built yet, or a system of that scale cannot be practically assembled for a few experiments. Typically, these targets for sustained floating-point rates are derived from previous experience on similar platforms, if such platforms exist.

Performance Diagnostic Metrics

While TTS is the single most important metric, it alone does not help users understand or optimize their HEC system for a particular application. Performance diagnostic metrics help users analyze and quantify the performance of an application on a specific architecture and are often not general across architectures. For example the vector operational ratio metric is important for vector computers, while the cache miss ratio is important for cache-based computers. Clearly no finite set of execution metrics captures all performance features for all HEC systems.

We have found, however, that good performance models are useful in predicting general performance on different platforms. For example, in a computationally expensive kernel of the algorithm, a particular communication, I/O or computational pattern can be extracted to give a general idea of performance. For example, several applications rely on distributed matrix multiplications. Knowing the layout of the distributed data and the computational requirements allows the researcher to develop an educated performance model including computational speed, communication, and I/O bandwidth. While not easy to create, performance models that delve into some detail and examine the algorithm closely, not only allow for additional information in procurement activities, they also allow the code developer to understand bottlenecks for their algorithms (even before they are implemented).

Methods for deriving system performance targets from actual or projected application requirements or other user needs [6]

The computational workloads and communication patterns for scientific applications vary dramatically, depending, in part, on the nature of the problem the applications are solving. Though diverse, the computation and communication requirements for these applications play a critical role in the design of next-generation architectures and software. These requirements when combined with scalable performance models provide powerful evidence to drive forward the design of new systems. Although several teams have investigated the scalability, architectural requirements, and inherent behavioral characteristics of various scientific applications [4,5], few such comprehensive, in-depth studies exist for contemporary scalable HEC applications.

Users are often intensely involved in the procurement process. They are consulted to help understand their current and future application requirements. A good example of this is [4], which is published every three years and contains descriptions of user capabilities today in addition to their expectations, given more computational capability. Many of the contributions also contain a blue-sky scenario: how much capability the user would need to do what they consider to be the next level of interesting, high impact science.

References

- [1] D.J. Kuck, *High performance computing: challenges for future systems*. New York: Oxford University Press, 1996.
- [2] L. Derose, M. Pantano, J.S. Vetter, and D.A. Reed, "Performance Issues in Parallel Processing Systems," in *Performance Evaluation - Origins and Directions*, G. Haring, C. Lindemann et al., Eds.: Springer-Verlag, 2000.
- [3] D.E. Culler, J.P. Singh, and A. Gupta, *Parallel computer architecture: a hardware software approach*. San Francisco: Morgan Kaufmann Publishers, 1999.
- [4] NERSC Greenbook.
- [5] J.S. Vetter and F. Mueller, "Communication Characteristics of Large-Scale Scientific Applications for Contemporary Cluster Architectures," Proc. International Parallel and Distributed Processing Symposium (IPDPS), 2002.